Task and Memory Mapping Optimization for SDRAM Interference Minimization on Heterogeneous MPSoCs

Alfonso Mascareñas González
ISAE-SUPAERO, Université de Toulouse
Toulouse, France
alfonso.mascarenas-gonzalez@isae-supaero.fr

Jean-Baptiste Chaudron
ISAE-SUPAERO, Université de Toulouse
Toulouse, France
jean-baptiste.chaudron@isae-supaero.fr

Frédéric Boniol
ONERA, Université de Toulouse
Toulouse, France
frederic.boniol@onera.fr

Youcef Bouchebaba
ONERA, Université de Toulouse
Toulouse, France
youcef.bouchebaba@onera.fr

Jean-Loup Bussenot
ONERA, Université de Toulouse
Toulouse, France
jean-loup.bussenot@onera.fr

***Abstract*—DDR SDRAM memories are resources commonly used on multicore platforms and hence, being a main source of interference. To deal with this issue, we propose a methodology based on task/memory mapping optimization through multi-objective heuristic-based algorithms. By placing the tasks on the platform cores and the memory in the DDR SDRAM banks, we minimize the DDR SDRAM interference while considering other aspects such as the task execution parallelism and deadline margin. To evaluate the fitness of the task/memory map, the optimization algorithms make use of cost function equations. In order to compute the DDR memory interference cost, we use a fast executing self-designed cost function. The execution parallelism is computed using the workload variance cost function. The deadline margin of a task is computed considering the inter and intra core interference. The task/memory mapping outcomes are checked through tests for which the heterogeneous MPSoCs Keystone II and Sitara AM5728 are used. To assure certification, the WCET constraints of the resulting near-optimal Pareto solutions are verified through formally validated bounding frameworks.**

***Index Terms*—Heterogeneous multicore platforms, Task and memory mapping, Multi-objective optimization, DDR SDRAM**

I. Introduction

The DDRx SDRAM memory is the main memory for the Processing Elements (PEs) making up a multi-core platform. Therefore, bounding and reducing the impact of the interference taking place on this memory is essential for real-time critical applications. For this purpose, task and memory mapping, which allows to allocate the tasks to dedicated cores as well as restricting its respective data to bounded DDR3 memory address regions, can be used. In addition to the DDR3 memory interference, some other criteria (e.g., parallelism, energy consumption [5]) can be optimized at the same time by using multi-objective optimization algorithms. These algorithms yield near-optimal mappings (in our case) for a platform to behave at its best according to the selected criteria and respecting some constraints (e.g., deadlines). To do so, they require a cost function per criterion to optimize. Multi-objective mapping based on metaheuristic algorithms, e.g., *Evolutionary Algorithms* (EA), are those multi-objective algorithms whose working principle is based on a heuristic. This allows to efficiently explore the search space at the expense of returning near-optimal solutions.

The aim of this work is to propose an efficient method based on near-optimal task/memory mapping that minimizes the worst-case DDR3 SDRAM interference while considering two other criteria (execution parallelism and margin deadline) through multi-objective heuristic-based optimization algorithms. The DDR memory interference impact is computed by a self-designed cost function equation which executes fast enough to be integrated into an optimization strategy. The execution parallelism of the tasks, which is evaluated through the load variance cost function, is chosen as a second conflicting optimization objective. As a third objective, we have chosen the task deadline margin which is computed considering the intra and inter core interference. Opposing the three of them yields a Pareto front which allows the designer to decide the near-optimal task/memory mapping that better suits their design (Section V). The developed task/memory mapping optimization tool makes use of the general-purpose Python metaheuristics-based framework for multi-objective optimization called jMetalPy [3]. The near-optimal mapping evaluations are done on the heterogeneous platforms Keystone II [9] and Sitara AM5728 [10] by Texas Instruments (TI). In order to carry out the previous evaluations, a set of tests and a measurement framework based on performance counters and time stamp counters are used. As the considered DDR memory interference cost function equation validation was done through measurements, it may be considered unsafe. Therefore, we propose to attach to our mapping tool a WCET verification system, which would determine if the resulting Pareto solutions are indeed safe or not. This verification

system would be made of a formally proven DDR memory interfering bounding method, e.g., [8]. The multicore platforms implementation, the DDR memory cost function and the task/memory mapping tool for heterogeneous platforms can be found in this repository[1].

II. Related Work

A. DDR Interference Analysis

DDR SDRAM bounding can be achieved via different approaches: (1) request-driven analysis, (2) job-driven analysis, (3) hybrid analysis and (4) holistic analysis. Request-driven focuses on the interference that an individual memory request receives. Afterwards, all the task request is multiplied by the interference cost [7], [14]. Job-driven is based on the number of interfering memory requests from other cores of the platform. Some works consider the minimum of the last two analysis in order to reduce the overestimation [11], [15]. The hybrid analysis computes the memory interference bounding by mixing the request-driven and the job-driven analysis [8]. The holistic analysis focuses on the system rather than its parts [6]. We remark the use of inequations and solvers to obtain the bounding like it is done in works [6], [8]. The use of inequations to obtain exact bounds cannot be integrated in our framework because of the complexity and time consuming of solving these. However, their use is proposed for checking the validity of the task/memory mapping solutions given by the metaheuristic algorithms.

B. Task Mapping Optimization

The use of metaheuristic algorithms for task mapping is a common practice for finding efficient solutions to multi-objective problems. [5] shows how through a self designed mapping tool called MPASSIGN, tasks are satisfactorily mapped on a NoC-based homogeneous many-core platform according to the selected objectives, e.g., workload and memory distribution, energy efficiency, NOC communication cost. Work [4] makes use of genetic algorithms for efficiently assigning tasks to nodes in a heterogeneous platform while dealing with multiple conflicting objectives. These objectives are: (1) the task execution quality maximization, (2) the energy and (3) bandwidth consumption minimization. Both works [4], [5] deal with multi-objective optimization through task mapping as intended in our work, but they do not target the DDR memory interference optimization which is our main goal. Nonetheless, the reduction of DRAM memory interference via task mapping has been already put into practice in works like [13]. In this work, the idea is to keep the memory demands of tasks concurrently executing below the minimum available memory bandwidth. Tasks are strategically placed on cores in order to minimize the main memory interference using three approaches: (1) A heuristic algorithm called FFDM that focuses on the memory demands, (2) two neighborhood search algorithms for improving the initial solution yielded by FFDM and (3) a metaheuritic-based simulated annealing which approximates global optimization. [11] designs their own task mapping algorithm based on the conclusions extracted during their DRAM analysis: DRAM bank partitioning and co-locating memory-intensive tasks on the same core reduces interference. In contrast to our work, [11], [13] focus on single-objective optimization, missing the opportunity to optimize conflicting objectives accompanying the DDR interference, e.g., parallelism.

[1]https://github.com/ISAE-PRISE/sinteo

III. Background

A. DDR3 SDRAM Device Addressing

In terms of address mapping, the main components making up a DDR3 memory device are the ranks, the banks and their associated row buffers. A rank is a collection of banks. A bank is a memory device defined by columns and rows where data is stored. Banks work independently from each other and in parallel, what reduces certain operational latencies. Attached to each bank there is a row buffer which stores a row of data. These buffers work similar to caches. Every time a new data row for a given bank needs to be manipulated, the current row present in the buffer must be saved before loading the new one.

The memory is accessed by defining the column, row, bank and rank values. From a grid perspective, the columns represent the Y axis, the rows the X axis, and the ranks and banks are regions and sub-regions of the grid (see Figure 1). By design, all the banks can be kept opened, allowing interleaving among them with no costs. However, switching to a different row for a given bank has a time penalty (row buffer precharge and activation). Similarly, switching among ranks has an impact. The physical address bits are used to select the column, row, bank and rank. On the Keystone II platform, the address decoding is done as follows: $address \rightarrow RowSize|RankSize|BankSize|ColumnSize|Bus\ Size$. In bits length, the resultant decoding would be: $address \rightarrow 16|0|3|10|3$, e.g., the physical address 0x80282000 generates an output like: column 0, row 32808, bank 1. It is by manipulating the latter (e.g., through the memory management unit, pointers, padding) that we can achieve the bank partitioning.

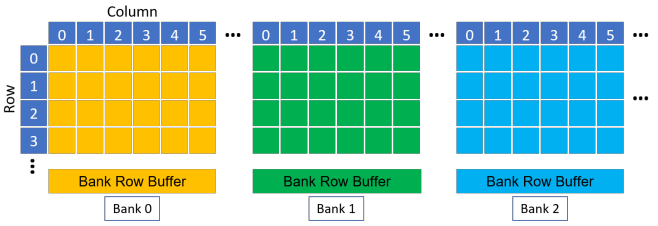


Fig. 1: DDR memory organization: Columns, Rows and Banks

B. Metaheuristic-based Multi-objective Optimization

Metaheuristic-based algorithms are high-level procedures that, in spite of mathematically not assuring the most optimal solution, offers great ones especially for NP-hard problems, e.g., task mapping. Among these algorithms we can distinguish those used for single-objective optimization, e.g.,

mono-objective simulated annealing, and for multi-objective optimization, e.g., Particle Swarm Optimization (PSO). This work focuses on the latter type of algorithms as we want to optimize several aspects at the same time. In this work, the algorithms used for task mapping are population-based algorithms. Inside this category, we differentiate among five types of EAs according to its heuristic: the genetic-based EA (NSGA-II, SPEA2 and MOCell), swarm-based (SMPSO and OMOPSO), differential-based EA (GDE3), decomposition-based EA (MOEA/D) and indicator-based EA (IBEA). These population-based EAs generally consist in generating an initial population, i.e., group of individuals, and modify the individuals of that population, i.e., set of solutions, according to the algorithm heuristic and the measured fitness using a cost function. For example, genetic algorithms, which are based on genetic representation, generates sets of solutions (chromosomes). These chromosomes contains all the variables (genes) to tune for finding the best combination. A chromosome interacts with another (parents) by exchanging genes (crossover) in order to generate new chromosomes (offspring). However, only the fittest chromosomes, based on the cost functions, are selected for breeding. This process ensures that the best solutions sets are kept and the worst are discarded. The genes of the offspring may randomly change (mutation) to avoid convergence. The process is repeated until the termination conditions are met, e.g., number of iterations. The different working principles of the metaheuristic algorithms condition their final output. According to works [2], [5], to find the most complete sets of solutions, different algorithms should be used for the same problem. Thus, all the aforementioned algorithms are used, but only a few of them are shown in the results in Section V-D due to visualization purposes.

C. System On a Chip

This work uses the TI heterogeneous multicore SOC **Keystone II** model TCI6636K2H [9] located inside the EValuation Module (EVM) TCIEVMK2H. The SOC is made up of different PEs, where we can point out (I) 4 ARM Cortex A15 cores, (II) 8 C66x DSP cores. Furthermore, the TI heterogeneous multicore SOC **Sitara AM5728** [10] found in the BeagleBoard X15 board is also used for the task mapping validation process (see Section V-D). This SOC is made up of many different processors, where the 2 C66x DSPs and 2 ARM Cortex A15 cores available are used. The SOC heterogeneity plays an important role on the task/memory mapping process. An DSP or ARM core's instruction pipelining, out of order execution (reservation stations size), speculative execution (branch predictor) or multithreading capabilities greatly defines the processors performance and its interfering capacity.

1) Considerations: Along this work the following points about the multicore platform usage always apply:

- **No data caches**: We disable the L1D and L2 cache to increase the task's access frequency to the DDR3 memory and just focus on one shared memory, i.e., no shared caches.
- **Bare-metal**: No operating systems have been used in order to gain entire control of the system and ease the data treatment due to OS derived effect, e.g., preemption, frequency throttling.
- **Priority and Starvation**: The interconnection slave priorities of the platforms are equally set to ensure the same treatment to every core (one priority level).
- **Periodicity**: Tasks are periodically executed using a non-preemptive Start-Finish-Idle pattern in a predefined order.

IV. Cost Functions Description

A. Task Model

The cost functions used in this work relies on different task properties. A task τ_i is described by:

$$\tau_i := (C_i, A_i, SP_i, S_i, ACOR_i, PE_i, B_i, T_i)$$

where:

- C_i: The Worst-Case Execution Time (WCET) in isolation. $C_i \, \epsilon \, \mathbb{N}^+$.
- A_i: The number of DDR3 accesses. $A_i \, \epsilon \, \mathbb{N}^+$.
- SP_i: The *Store Proportion* (SP) indicates the share of stores in A_i. $SP_i = 1 - LP_i$ where SP_i and $LP_i \, \epsilon \, [0, 1]$. LP_i is the *Load Proportion* of A_i.
- S_i: The number of row switches in isolation. $S_i \, \epsilon \, \mathbb{N}^+$.
- $ACOR_i$: The *Average Commands per Opened Row* (ACOR) defines the number of DDR commands that τ_i can execute before a bank row switch is produced by another task on a different PE pointing to the same bank. $ACOR_i$ is 1 for closed-page policy SDRAM controllers.
- B_i: The bank to which τ_i is mapped to. $B_i \, \epsilon \, [0, N_b - 1]$.
- PE_i: The PE to which τ_i is mapped to. $PE_i \, \epsilon \, [0, N_c - 1]$.
- T_i: The period of τ_i, which is also the deadline. $T_i \, \epsilon \, \mathbb{N}^+$.

The set of tasks is defined as $\mathcal{T} = \{\tau_0, ..., \tau_{n-1}\}$. Task properties C_i, A_i, SP_i, LP_i, S_i and $ACOR_i$ are measurement-based while B_i, PE_i and T_i are application design dependent.

B. Cost Functions

In this work, we make use of three cost functions: **Load variance**, **maximum DDR SDRAM interference** and **minimum deadline margin**. The **load variance** is used for calculating the distribution of the workload across the cores by considering the variance of the tasks execution time C_i as shown in Equation 1:

$$Workload Variance(\mathcal{T}) = \sum_{p=0}^{P-1} \frac{(load(p) - \bar{C})^2}{P} \quad (1)$$

where $load(p) = \sum_{PE_i = p} C_i$, $\bar{C}$ is the average load, and P is the number of considered PEs of the specified platform (Cortex A15 cores plus C66X DSPs in our case).

The **maximum DDR SDRAM interference** cost function yields the highest interference impact suffered by a task of the set. This function differentiates between the interference caused from the inside of the analyzed task DDR memory bank

(intra-bank interference) and the other banks (inter-bank interference) as DDR3 memory timings are not homogeneously applied. Therefore, we make use of the bank partitioning technique during the implementation on the platforms. The cost function considers the effect of command batching and the opened-row policy. As well, a differentiation of the DDR memory request types (e.g., read, write, row switch) is done (see Section IV-A and Table I). Due to space constraints, the development and evaluation of the self-developed cost function is omitted[2]. Thus, we jump directly to Equation 2, which computes the interference cost for a given task through the addition of the intra-bank ($IC^{intra}(\tau_i)$) and inter-bank ($IC^{inter}(\tau_i)$) interference.

$$IC(\tau_i) = IC^{intra}(\tau_i) + IC^{inter}(\tau_i) \tag{2}$$

It must be noted that the tasks will see their execution time enlarge after the interference insertion. This affects the proportion of time tasks interfere each other, subsequently affecting the interference impact which must be recalculated. To cope with this problem, we consider a recursive implementation of Equation 2 described by Algorithm 1. *IC* is repeatedly called until the DDR memory interference value converges. An array with the *IC* values for all tasks is returned.

Algorithm 1: recursive_IC_calculation

Input: $\mathcal{T}$ (input task set)
Output:
– $IC[\mathcal{T}]$ (interf. cost for each task in $\mathcal{T}$)
– B (true iff the algorithm succeeds)
Local Variables:
– n (the recursion index)
– $IC^n[\mathcal{T}]$ (interference cost at step n)

```
/* Initialization                                   */
n=0
∀τ_i ∈ T, IC^n[τ_i] = 0
/* Recursive loop                                   */
while true do
    for each τ_i ∈ T, compute IC^{n+1}(τ_i) by Eq. 2
    if ∀τ ∈ T, IC^{n+1}(τ_i) == IC^n(τ) then
        return (IC^n, true)
    if ∃τ_i ∈ T such that C_i + IC^{n+1}(τ_i) > T_i then
        return (IC^n, false)
    n = n + 1
```

Finally, the **maximum DDR SDRAM interference** cost function is defined by Equation 3 as the maximum DDR SDRAM interference cost from the set retrieved from Algorithm 1:

$$Maximum\,DDR\,SDRAM\,Interference\,(\mathcal{T}) = \max_{\tau_i \epsilon \mathcal{T}}(Algo1(\mathcal{T})) \tag{3}$$

The last cost function is the **minimum deadline margin** which is defined by Equation 4. This cost function retrieves the smallest time difference between the task deadline (period T_i in our case) and the total amount of time it takes to execute from the task set $\mathcal{T}$. The time a task takes to execute in the worst-case is assumed to be the WCET of the tasks on the same core ($C_i + \sum_{PE_j=PE_i}^{i \neq j} C_j$) plus the suffered DDR memory interference (Algorithm 1 ($\mathcal{T}$)).

$$Minimum\,deadline\,margin\,(\mathcal{T}) = \min_{\tau_i \epsilon \mathcal{T}}\{T_i - C_i - \sum_{PE_j=PE_i}^{i \neq j} C_j - IC_i\} \tag{4}$$

where IC_i is the interference cost returned by $Algo1(\mathcal{T})$ for task τ_i

V. Task and Memory Mapping Optimization

A. Task/Memory Mapping Tool

A task/memory mapping tool is developed in Python in order to explore the different mapping solutions on the Keystone II and Sitara AM5728 platforms. This aforementioned tool makes use of the jMetalPy framework [3], which provides a general-purpose multi-objective optimization environment and a large set of defined metaheuristic algorithms. Its design makes it easy to use for task/memory mapping purposes. Our tool considers the following features:

- ☐ **Exploration space**: The upper and lower bounds for the mapping solutions are set. These bounds are not homogeneous as the mapping solution contains the task-core and core-DDR memory bank maps, which differ from each other. Table II shows an example of mapping. For the task-core part, the value represents the core and the index the task. For the core-bank part, the index now represents the core and the value the DDR memory bank used by the tasks of that core. The blue and green colors represent the core type, e.g., Cortex A15, C66x DSP.
- ☐ **Cost functions**: The considered cost functions for the metaheuristic algorithms are added (see Section IV-B). The algorithms compute the near-optimal task-core and core-DDR3 memory bank maps according to these functions output.
- ☐ **Tasks**: The tasks τ_i to map and its profiling information (see Sections IV-A and V-B). This information is required by both, the problem definition and the cost functions.
- ☐ **Platform architecture**: The platform architecture specifications, e.g., PEs types and quantity, DDR memory layout, timings and thresholds, platform topology.
- ☐ **Constraints**: Mapping constraints are defined e.g., tasks total memory must fit inside the bank they are pointing to, deadlines must be respected.

B. On-Board Measurement Framework

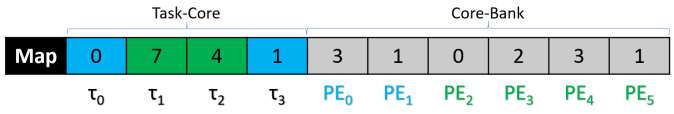
Metrics are used for: (1) tasks profiling, more specifically for C_i, A_i, SP_i, LP_i, S_i and $ACOR_i$ and (2) validation of the task/memory mapping. The amount of measurements used for (1) and (2) is 10^5. The measurements are obtained from the Keystone II and Sitara AM5728 in different ways depending on the target:

[2]Please, refer to the technical report on the development of the cost function at https://github.com/ISAE-PRISE/sinteo

TABLE I: Tasks main properties on Keystone II

Task	Type	Description	C (Cycles)	A (Accesses)	SP	LP	S (Actives)	ACOR
sb0 (ARM)	Synthetic benchmark	Integer vectors operations.	36315	437	0.312	0.688	0	2.46
sb0 (DSP)			33202	443	0.298	0.702	0	1.70
rb0 (ARM)	Real benchmark	ROSACE engine management.	3107	51	0.628	0.372	16	3.40
rb0 (DSP)		Floating-point operations.	12698	197	0.518	0.482	16	2.20
rb1 (ARM)	Real benchmark	ROSACE altitude filtering.	5110	84	0.631	0.369	14	3.36
rb1 (DSP)		Floating-point operations.	18927	304	0.514	0.486	23	2.10

TABLE II: Task/Memory mapping representation example



- **ARM Performance Monitor Unit**: The ARM cores make use of performance counters to measure events from a running program. The number of counters and events vary depending on the ARM architecture version. An ARM Cortex A15 core (architecture version 7) has a cycle execution counter and 6 general counters where we can set the event to study from a list (see [1]). These counters are accessed using a Start-Stop pattern.
- **DSP Time Stamp Counter**: The DSPs make use of a 64bit register to record the execution time. To avoid inconsistencies while retrieving the time value, a copy of the 32 MSBs is automatically triggered each time we access the 32 LSBs of the register. A Start-Read access pattern is used.
- **DDR3 Memory Controller Performance Counters**: The DDR3 controller provides two general purpose performance counters and one dedicated to the controller execution time. The events, e.g., accesses, reads, row actives, can be counted either from a specific core or a system perspective. The counters registers are accessed using a Start-Read pattern.

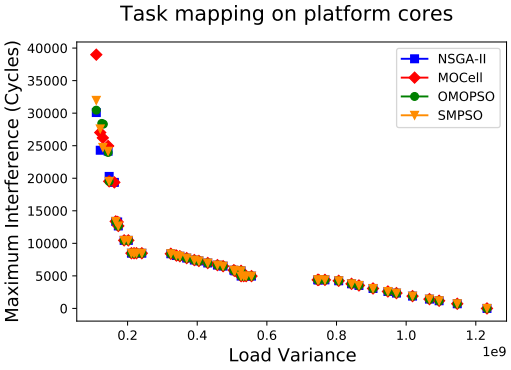
C. Testing Considerations

Benchmark tasks are used for making up the application whose tasks and associated memory are to be mapped by the mapping tool. We can distinguish two types of benchmarks: synthetic and real application tasks. Table I lists the tasks and their main characteristics for the Keystone II platform. The real application tasks are adaptations from the ROSACE case study tasks [12]. The application whose near-optimal maps are to be found by the mapping tool, is made up of 8 monoperiodic tasks (τ_0(rb1), τ_1(rb1), τ_2(rb1), τ_3(rb1), τ_4(sb0), τ_5(sb0), τ_6(rb0) and τ_7(rb0)) with a period of 1.2ms and a deadline equivalent to it (see tasks in Table I). Tests in Section V-D consider 4 SDRAM banks while tests in Section V-E consider 4 and 8 banks. For evaluation purposes, 8 cores out of the 12 available on Keystone II are used (2 ARM cores and 6 DSPs). The 2 ARM cores and 2 DSPs on Sitara are used.

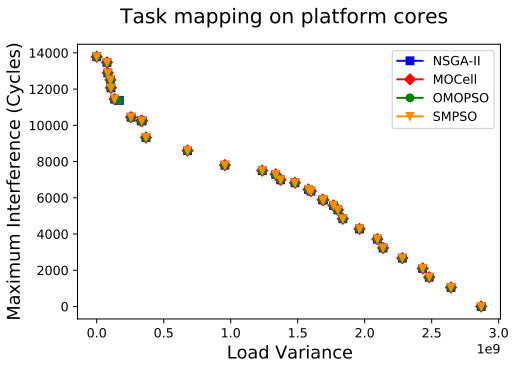
D. Two-Objective Optimization

The put into practice of the multi-objective algorithms for the task mapping is shown here. The Task/Memory Mapping Tool (see Section V-A) is used for retrieving the near-optimal mapping. The task mapping objectives are to minimize both, the DDR3 worst-case memory interference (*minimize(Maximum DDR SDRAM Interference($\mathcal{T}$))*) and load variance (*minimize(Workload Variance($\mathcal{T}$))*) of the tasks. Optimal and non-optimal sets of solutions are obtained when running the heuristic algorithms for the Keystone II and Sitara platforms considering the application described in Section V-C. All dominated solutions (non-optimal) are removed, leaving a set of solutions called Pareto frontier, where the involved trade-off is seen. Figures 2a and 2b show different Pareto frontiers from the set of algorithms used for the Keystone II and Sitara AM5728 respectively. Each different color depicts the non-dominated solutions for a particular algorithm according to its DDR3 memory interference (Y axis) and workload distribution (X axis). The different curves are the respective Pareto frontiers. In the top-left of Figure 2a, where the number of cores exceed the number of banks, we can appreciate heterogeneous solutions due to the difficulty of the meta-heuristic algorithms to find the best bank sharing combinations. This heterogeneity is not found in Figure 2b because SDRAM banks are never shared by cores. Finally, Figures 3a and 3b depict a unique Pareto frontier by removing the non-optimal solutions considering all the algorithms nondominated outputs for the Keystone II and Sitara AM5728 respectively. This set of solutions are the ones to be considered during our mapping. It is up to the designer to choose whether to enhance one feature at the expense of the other one (interference or parallelism). The first solution from the right with a maximum interference of zero is the monocore implementation where the interference is minimum and the variance maximum.

To check that these solutions are right, they are tested on their respective platforms. The five solutions marked in Figure 3a (I to V) are the ones chosen for evaluation on the Keystone II. They have been selected in maximum interference descendent order (from left to right). Their respective mapping configurations are shown in Table III, which follows the same logic as Table II. Cores 0 and 1 are ARM cores and from 2 to 7 DSPs. The resulting DDR interference impact for those five solutions are compared in Figure 4a. The X axis represents the different solutions while the Y axis shows the worst-case measured interference impact. The tasks whose interference impact was measured are shown in green, red, yellow and blue colors which belong to tasks τ_0, τ_4, τ_5 and τ_7 respectively. We should pay special attention to τ_4 and τ_5 which are the larger tasks in terms of execution time and DDR accesses and hence the most interfered ones. Solution I has all its tasks



(a) On Keystone II.



(b) On Sitara AM5728.

Fig. 2: Multiple algorithms nondominated set of solutions.

TABLE III: Task maps to validate on Keystone II

	Task-Core Map	Core-Bank Map	Cores Used	Banks Used
I	1 1 0 0 4 7 1 6	3 3 1 0 1 2 0 2	5	4
II	1 1 0 0 2 6 1 0	2 3 1 3 1 3 0 3	4	4
III	0 0 1 0 0 7 1 1	2 3 0 3 3 1 2 0	3	3
IV	1 0 1 0 5 5 2 1	0 3 0 0 3 2 0 1	4	4
V	0 0 1 1 5 5 5 1	3 1 3 3 3 1 2 0	3	3

TABLE IV: Task maps to validate on Sitara AM5728

	Core Map	Bank Map	Active Cores	Banks Used
I	2 1 2 0 1 0 3 3	1 3 2 0	4	4
II	0 1 3 1 0 1 2 0	1 0 2 3	4	4
III	1 0 1 0 1 0 1 0	1 2 2 2	2	2
IV	1 0 0 1 1 1 2 0	0 1 2 0	3	3
V	1 1 0 0 1 1 1 1	1 0 3 0	2	2

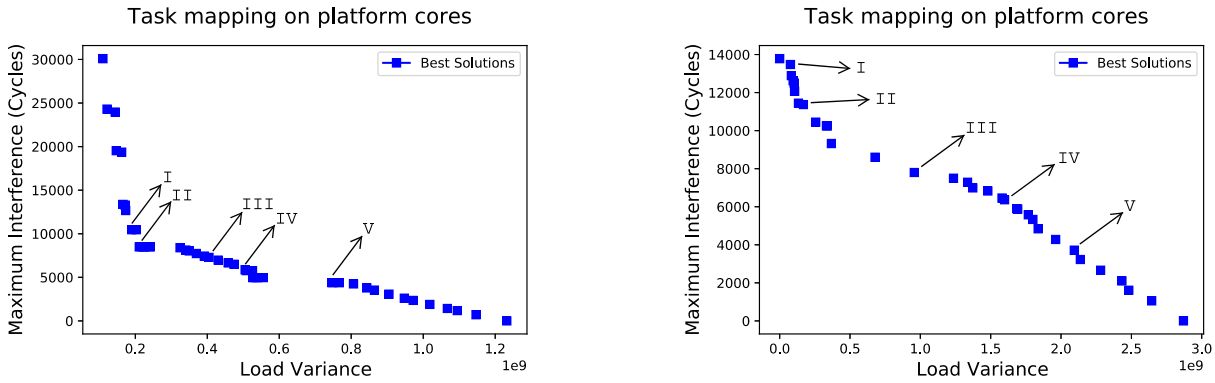
severely affected compared to the other solutions, which is caused by the bank sharing (5 cores against 4 DDR3 banks). Solutions II, III, IV and V are good examples for showing that the number of cores not always mean higher maximum interference, especially when each core has a private bank. Solutions II and IV have more overall interference due to an extra core running in parallel. However, Solution III, which uses three cores, splits τ_4 and τ_5 (most interfering tasks) onto two cores, reducing the workload variance to the detriment of the maximum interference. Therefore, through the spreading of execution intensive tasks we reduce the workload variance and through the stacking of these onto the same core we reduce the DDR3 interference. Solution V is the lowest interfered due to the τ_4 and τ_5 merge and the use of three cores.

Another measurement-based validation test is provided using the Sitara SoC. Figure 3b depicts the near-optimal Pareto frontier solutions found for the same set of tasks previously used. Nevertheless, the profiling values for these change as they are executed on another system. Compared to the Keystone II configuration, here the cores have always private banks. Table IV shows the mapping of the marked solutions I to V. The cores 0 and 1 are ARM cores and 2 and 3 are DSPs. Banks from 0 to 3 are used. Figure 4b shows the measured data of the tagged solutions in descendent order of interference. It can be seen how the worst case interference, found in $\tau 4$ or $\tau 5$, is reduced. As per Figure 3b, note again that a solution with less cores can produce more interference than one with more (Solutions III and IV). The reason is the same as on the Keystone II: the most interfering tasks split on two cores. Solution III mapping separates $\tau 4$ and $\tau 5$ on two different cores. In this way, $\tau 4$ is interfered during its whole execution by $\tau 5$ and vice-versa. Solution IV groups $\tau 4$ and $\tau 5$ together, which are interfered during 42% of their execution by one core and 39% by the other one.

In terms of execution time, the solutions for algorithms NSGA-II, MOCell, OMOPSO, SMPSO in Figure 3a were obtained in 9.29, 3.29, 3.28 and 2.52 minutes respectively with an algorithm iteration speed of 63.23, 167.23, 172.61 and 208.61 iterations/s. The same algorithms took 9.07, 2.59, 3.28 and 2.08 minutes for solutions in Figure 3b with a speed of 65.57, 195.46, 255.85 and 279.44 iterations/s. The heuristic algorithms were configured to carry out 35000 evaluations (iterations) with a population size of 2000. A computer with an Intel Core i7-8750H CPU @ 2.20GHz was used.

E. Three-Objective Optimization

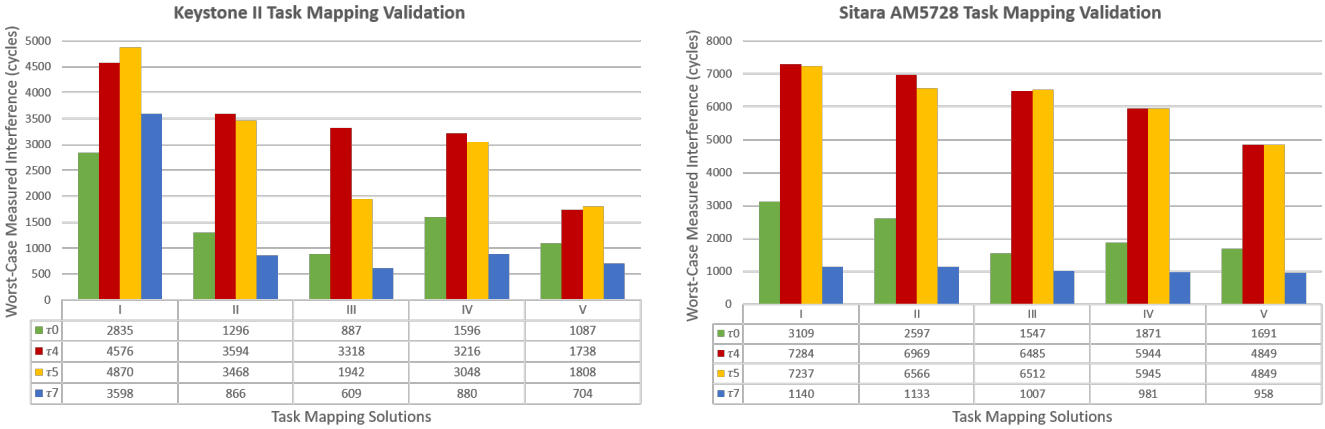
In here, the deadline margin is added as a third objective in order to increase the gap between the deadline and the WCET of a task after interference. The objective is defined as *maximize(Minimum Deadline Interference($\mathcal{T}$))*. This objective is positively correlated with the execution parallelism while it is negatively correlated with the SDRAM interference. The results are presented in Figures 5a and 5b where 4 and 8 DDR SDRAM banks are used. Both figures shows the same kind of behavior. From closer to farther, the mapping solutions see their load variance decrease. Consequently, the minimum deadline margin difference and maximum DDR memory interference increase. However, there is a change of behavior that implies to decrease the deadline margin and increase the memory interference in exchange of less load variance. It is in this last behavior where Figures 5a and 5b differ from each other. The former is more affected in terms of interference due to the DDR memory bank sharing among the cores. The latter uses 8 banks, and hence, every core can have its own private bank. This leads to an interference reduction.



(a) On Keystone II.

(b) On Sitara AM5728.

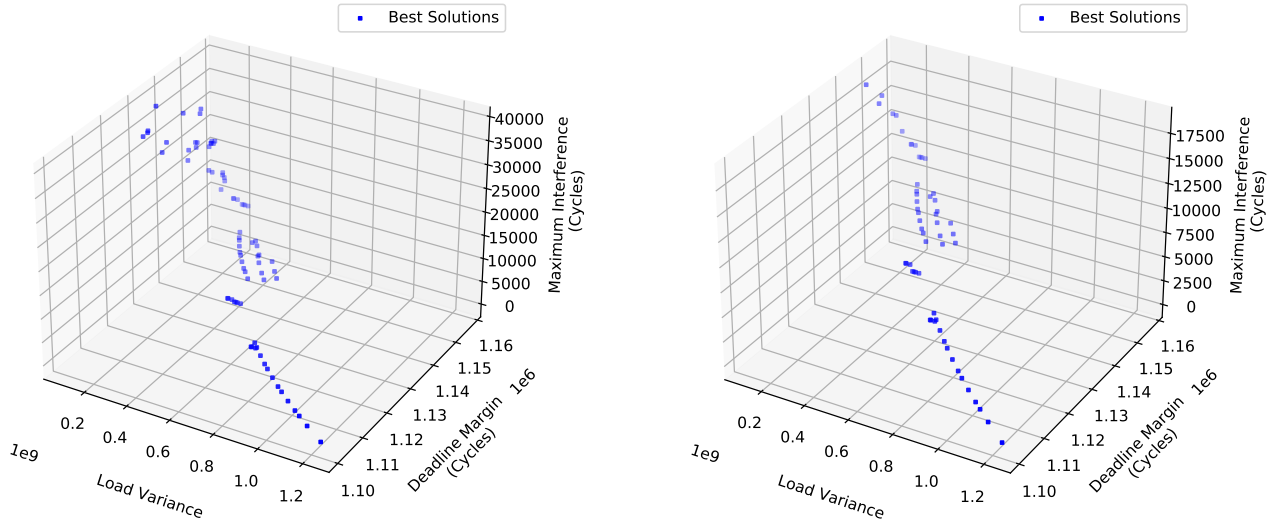
Fig. 3: Absolute nondominated solutions.



(a) On Keystone II.

(b) On Sitara AM5728.

Fig. 4: Measurement-based task mapping validation.



(a) Assuming 4 DDR memory banks.

(b) Assuming 8 DDR memory banks.

Fig. 5: Three-objective task/memory mapping optimization on Keystone II.

F. WCET Verification

In Sections V-D and V-E, we show the Pareto near-optimal solutions for minimizing the maximum DDR memory interference and workload variance, and maximizing the deadline margin through multi-objective optimization algorithms. However, as the used interference cost function had only been proven based on measurements (not formally proven), a verification step is required to ensure that the WCET constraints are satisfied. This verification step, which is required for certification, consists in checking if the real WCET of the yielded tasks remain below their respective deadlines according to other DDR memory bounding methodologies such as the inequations-based bounding through linear optimization proposed in [6], [8]. Their methods are formally proven, meaning that the maximal interference value they return is an upper-bound of the real interference experienced by the tasks. Solving inequations is time expensive if used for the solutions exploration but not when checking the near-optimal solutions. The diagram found in Figure 6 describes how would be the entire procedure. The blue block at the left represents the task/memory mapping process. Its output (Pareto front) is passed as an input to the WCET validation block in green color. In this block we run a verified bounding methodology for each solution of the Pareto front. Those solutions whose tasks do not comply with their imposed deadline are removed.

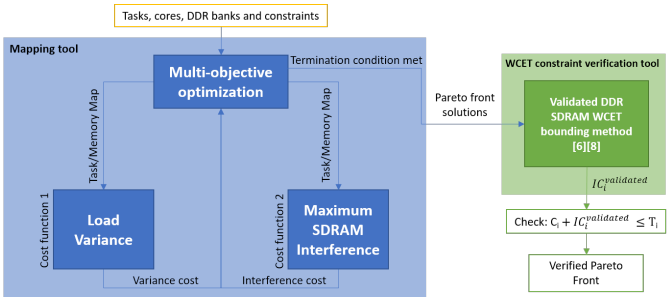


Fig. 6: Overview of the task/memory mapping system with formal verification.

VI. Conclusions and Future Work

We show the capabilities of task/memory mapping regarding the retrieval of near-optimal DDR SDRAM interference solutions while optimizing conflicting objectives. It could be seen the trade-off that implied minimizing the SDRAM interference and the task workload variance on the Keystone II and Sitara AM5728 SoCs (see Figures 3a and 3b) and the effect of maximizing the task deadline margin on the Keystone II platform (see Figures 5a and 5b). This allows us to select the solution that better complies with our application constraints, e.g., timing, safety margin, interference. We remark the fact that there are solutions where less cores concurrently working lead to more interference. This is due to the tasks stacking on the cores and the core type. Besides, Pareto solutions avoid sharing banks due to the high intra-bank cost, unless it is strictly necessary, i.e., the number of cores is bigger than the number of banks.

During this work data caches have been disabled to increase the DDR3 memory access rate and avoid shared caches interference. As future work, we would like to analyse this kind of interference and the privatization of shared caches from the task/memory mapping perspective. Besides, it would be interesting to reproduce this presented work within a RTOS environment. Moreover, the proposed WCET verification system implementation is pending.

Acknowledgment

This work was supported by the Defense Innovation Agency (AID) of the French Ministry of Defense (research project CONCORDE N° 2019 65 0090004707501) and the French Civil Aviation Authority (DGAC) (research project PHYLOG).

References

[1] ARM. *Cortex-A15 MPCore. Technical Reference Manual*, March 2012.

[2] Dihia Belkacemi, Youcef Bouchebaba, Mehammed Daoui, and Mustapha Lalam. Network on chip and parallel computing in embedded systems. In *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*, pages 146–152, 2016.

[3] Antonio Benítez-Hidalgo, Antonio J. Nebro, José García-Nieto, Izaskun Oregi, and Javier Del Ser. jmetalpy: A python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, page 100598, 2019.

[4] Carolina Blanch Perez del Notario, Rogier Baert, and Maja D'Hondt. Multi-objective genetic algorithm for task assignment on heterogeneous nodes. *International Journal of Digital Multimedia Broadcasting*, 01 2012.

[5] Youcef Bouchebaba, Pierre Paulin, Ali Erdem Ozcan, Bruno Lavigueur, Michel Langevin, Olivier Benny, and Gabriela Nicolescu. Mpassign: A framework for solving the many-core platform mapping problem. In *Proceedings of 2010 21st IEEE International Symposium on Rapid System Protyping*, pages 1–7, 2010.

[6] Daniel Casini, Alessandro Biondi, Geoffrey Nelissen, and Giorgio Buttazzo. A holistic memory contention analysis for parallel real-time tasks under partitioned scheduling. pages 239–252, 04 2020.

[7] Mohamed Hassan and Rodolfo Pellizzoni. Bounding dram interference in cots heterogeneous mpsocs for mixed criticality systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2323–2336, 2018.

[8] Mohamed Hassan and Rodolfo Pellizzoni. Analysis of Memory-Contention in Heterogeneous COTS MPSoCs. In *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, volume 165, pages 23:1–23:24, 2020.

[9] Texas Instruments. *66AK2Hxx Multicore DSP+ARM® KeyStone™ II System-on-Chip (SoC)*, October 2017.

[10] Texas Instruments. *AM572x Sitara Processors - Technical Reference Manual*, August 2019.

[11] Hyoseung Kim, Dionisio de Niz, Björn Andersson, Mark H. Klein, Onur Mutlu, and Ragunathan Raj Rajkumar. Bounding and reducing memory interference in cots-based multi-core systems. *Real-Time Systems*, 52:356–395, 2016.

[12] Claire Pagetti, David Saussié, Romain Gratia, Eric Noulard, and Pierre Siron. The rosace case study: From simulink specification to multi/many-core execution. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium*, pages 309–318.

[13] Syed Aftab Rashid. Server based task allocation to reduce inter-task memory interference in multicore systems. In *2019 International Conference on Frontiers of Information Technology*, pages 322–3225.

[14] Zheng Wu, Rodolfo Pellizzoni, and Danlu Guo. A composable worst case latency analysis for multi-rank dram devices under open row policy. *Real-Time Systems*, 52, 11 2016.

[15] Heechul Yun, Rodolfo Pellizzon, and Prathap Kumar Valsan. Parallelism-aware memory interference delay analysis for cots multicore systems. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 184–195, 2015.